the execution of the operation by returning the insertion operation to task queue 260. Returning the operation to task queue 260 causes the operation to be retried during the next update phase of step 130 (FIG. 1). Computer code 220 then processes the next operation that was received in step 510.

**[0041]** If on the other hand it is determined in step 586 that the insertion operation can execute without conflicts, then in a step 592 the flag written in step 582 is modified to indicate that conflict tests have been checked. Similarly, the flag written in step 580 is modified to indicate that conflict tests have been checked. Note that although no conflict tests were performed following step 580, the flag written in step 580 is thus far the first data 250 associated with a particular row 320 and, hence, there can be no conflicts.

**[0042]** In a step 594 computer code 220 creates new links 450 and 460 pointing from first element 420 to new element 410 and from new element 410 to element 430 respectively. Links 450 and 460 are only visible to the process performing the insertion of new element 410. In a step 596 computer code 220 proceeds to the next operation that was received in step 510.

**[0043]** Once all update phase state transition of operations received in step 510 are handled, then the update phase of step 130 (FIG. 1) is completed.

**[0044]** FIG. 6 illustrates the process steps of the insertion operation which occur during the commit phase of step 140 (FIG. 1) according to the invention. In a step 620 computer code 220 uses a first pointer 315 in RID 310 to navigate to the first row 320. In a decision step 630 PU 330 of the first row 320 is examined. If PU 330 is found to have a non-NULL value indicating that update data 250 for the first row 320 exists, computer code 220 proceeds to a step 640 in which top item 340 of update data 250 is read. In the exemplary case, PU 330 is non-NULL and computer code 220 then uses the read data in a step 650 to transition the state of element 410 being inserted from the pending insert state to the valid state. The data read, including the pointer to the insertion point, is used to complete the insertion of new element 410 by eliminating link 440 and making links 450 and 460 visible to all processes operating upon data structure 240.

**[0045]** Commit phase state transitions of additional operations received in step 510 are executed by reading and executing, in order, any remaining items of update data 250 associated with the first row 320. Upon execution of all items in such update data 250, the commit phase of step 140 determined in decision step 660 if all pointers 315 in RID 310 have been processed. Step 660 is executed directly after step 630 if the PU 330 examined in step 630 includes a NULL pointer. If all pointers

315 in RID 310 have been processed the method proceeds to step 120 (FIG. 1). Otherwise, the next pointer 315 is used in a step 670, analogous to step 620 to navigate to subsequent rows 320 where the associated PU 330 is examined as described herein.

**[0046]** While the update and commit phases of steps 130 and 140 (FIG. 1) have been described with reference to an insertion operation, any operation upon data structure 240 can be divided into the two phases described. Many multi-step operations, such as sorting elements 245, balancing of tree data structures, and the like, are combinations of insertions, and deletions, of elements 245 and can therefore be executed in the update and commit phases of steps 130 and 140.

**[0047]** The system and method of the invention find particular implementation in the rebalancing of tree data structures. Conventional systems either rebalance a tree data structure with every insertion or deletion of a node that causes the tree to become unbalanced, or allow the tree data structure to become unbalanced.

**[0048]** As is well known in the art, searches or traversals of an unbalanced tree are more costly than such operations on a balanced tree. This inefficiency has motivated the prior art to rebalance whenever the tree becomes unbalanced, however slightly.